RIDE: Reversal Invariant Descriptor Enhancement

Lingxi Xie[1*] Jingdong Wang[2] Weiyao Lin[3] Bo Zhang[4] Qi Tian[5]
[1,4]LITS, TNList, Dept. of Comp. Sci. & Tech., Tsinghua University, Beijing, China
[1]Department of Statistics, University of California, Los Angeles, Los Angeles, LA, USA
[2]Microsoft Research, Beijing, China
[3]Department of Electronic Engineering, Shanghai Jiao Tong University, Shanghai, China
[5]Department of Computer Science, University of Texas at San Antonio, San Antonio, TX, USA
[1]198808xc@gmail.com [2]jingdw@microsoft.com
[3]wylin@sjtu.edu.cn [4]dcszb@mail.tsinghua.edu.cn [5]qitian@cs.utsa.edu

Abstract

In many fine-grained object recognition datasets, image **orientation** *(left/right) might vary from sample to sample. Since handcrafted descriptors such as SIFT are not reversal invariant, the stability of image representation based on them is consequently limited. A popular solution is to augment the datasets by adding a left-right reversed copy for each original image. This strategy improves recognition accuracy to some extent, but also brings the price of almost doubled time and memory consumptions.*

In this paper, we present **RIDE** *(Reversal Invariant Descriptor Enhancement) for fine-grained object recognition. RIDE is a generalized algorithm which cancels out the impact of image reversal by estimating the orientation of local descriptors, and guarantees to produce the identical representation for an image and its left-right reversed copy. Experimental results reveal the consistent accuracy gain of RIDE with various types of descriptors. We also provide insightful discussions on the working mechanism of RIDE and its generalization to other applications.*

1. Introduction

Image classification is a fundamental problem in computer vision which implies a large number of applications. One of the most popular approaches for image classification is the Bag-of-Features (BoF) model [8], a statistics based algorithm in which local features are extracted, encoded and summarized into global image representation.

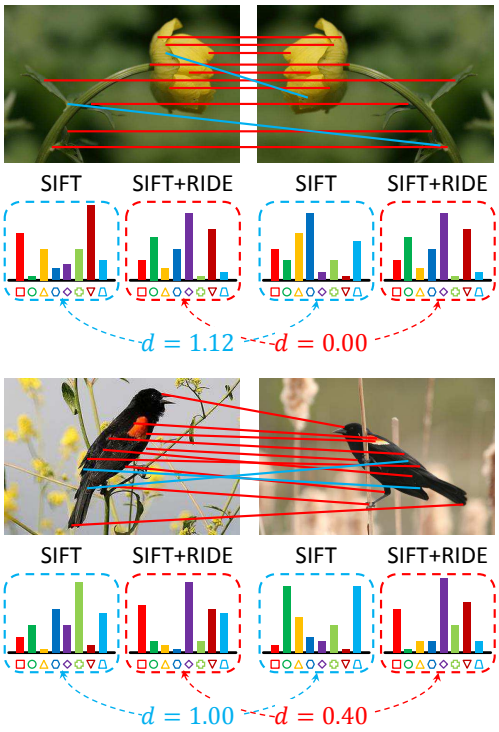

Figure 1: SIFT [21] matching with (red) and without (blue) RIDE (best viewed in color). In the latter case, it is difficult to find feature matches even between an image and its reversed copy. RIDE also significantly reduces the feature distance between each pair of reversed objects.

Recent years have witnessed a shift in the interest towards fine-grained recognition, which is aimed at predicting the class at a finer level of granularity. For example, given that each image contains a *bird*, it remains to decide

*This work was done when Lingxi Xie was an intern at MSR. This work is supported by the 973 Program of China 2013CB329403 and 2012CB316301, NSFC 61332007, 61273023, 61429201, 61471235, Tsinghua ISRP 20121088071, ARO grants W911NF-15-1-0290 and W911NF-12-1-0057, and Faculty Research Awards, NEC Lab of America.

which species is depicted. As observed in [3][6][12], the key to fine-grained recognition is the alignment of semantic object parts, such as the *head* or *tail* of a *bird*. Meanwhile, we also observe that image reversal harms the object alignment quality, as illustrated in Figure 1. Since many handcrafted descriptors, such as SIFT [21] and LCS [29], might change completely after being reversed, it is difficult to find feature correspondence between an image and its reversed version. Consequently, the vector representation of an image might be totally different from that of its reversed version, which limits the machine learning algorithms from learning discriminative models. To cope with, researchers propose to perform classification on an augmented dataset, which is constructed by adding a reversed copy for each image [7][6]. Although such algorithms improve recognition accuracy to some extent, they still suffer the disadvantage that requiring almost doubled computational costs.

This paper presents **RIDE** (Reversal Invariant Descriptor Enhancement), a simple, fast and generalized algorithm which brings reversal invariance to local descriptors. We start with observing the difference between original and reversed descriptors, and then suggest computing the orientation of each descriptor to cancel out the impact of image reversal. For orientation estimation, we adopt an approximated summation on the gradient-based histograms. When RIDE is adopted with the BoF model, we guarantee to generate identical representation for an image and its left-right reversed copy. Experiments reveal that RIDE produces consistent accuracy improvement on a wide range of fine-grained object recognition and scene classification tasks. RIDE even beats the data augmentation methods with higher recognition rates and lower time/memory consumptions. The easy implementation and cheap computational costs make RIDE quite competitive in real-world applications.

The remainder of this paper is organized as follows. Section 2 briefly introduces related works. The RIDE algorithm and its application are illustrated in Section 3. After experiments are shown in Section 4, we conclude in Section 5.

2. Related Works

2.1. The BoF Model

The BoF model [8] starts with extracting local descriptors. Due to the limited descriptive power of raw pixels, handcrafted descriptors, such as SIFT [21][36], HOG [9] and LCS [29], are widely adopted. Although these descriptors could be automatically detected using operators such as DoG [21] and MSER [24], the dense sampling strategy [4][35] works better on classification tasks.

Next, a visual vocabulary (codebook) is trained to estimate the feature space distribution. The codebook is often computed with iterative algorithms such as K-Means or GMM. Descriptors are then encoded with the codebook. Popular feature encoding methods include hard quantization, sparse coding [49], LLC encoding [39], super-vector encoding [56], Fisher vector encoding [33], etc.

In the final stage, quantized feature vectors are aggregated as compact image representation. Sum pooling, max-pooling and ℓ_p-norm pooling [11] provide different choices, and visual phrases [53][45] and/or spatial pyramids [13][19] are constructed for richer spatial context modeling. The representation vectors are then summarized [47] and fed into machine learning algorithms such as the SVM.

2.2. Towards Reversal Invariance

One of the major shortcomings of the BoF model comes from the sensitivity of local descriptors. Especially, in fine-grained recognition, objects might have different left/right orientations. Since handcrafted descriptors such as SIFT are not reversal invariant, feature representation of an image and its reversed version might be totally different.

To cope with, researchers propose to augment the image datasets by adding a reversed copy for each original image, and perform classification on the enlarged training and testing sets [7][6]. In [28], it is even suggested to learn a larger image transformation set for dataset augmentation. Although these complicated algorithms are verified to improve recognition accuracy, they still suffer the disadvantage of expensive time and memory overheads.

There are also efforts on designing reversal invariant descriptors for image retrieval. Some of them [22][46] consider geometry-inverted and brightness-inverted variants, and perform a symmetric function, such as dimension-wise summation or maximization, to cancel out reverse. Other examples include defining a set of spatial bins to calculate histograms [15], or enforcing that the flows of all regions should follow a pre-defined direction [55]. These inspire us that symmetry is the key to reversal invariance [34][40].

3. The RIDE Algorithm

This section presents the **RIDE** algorithm which brings reversal invariance to local descriptors.

3.1. Why Reversal Invariance?

In almost every fine-grained image collection, there exist both left-oriented and right-oriented objects. For example, among 11788 images of the **Bird-200** dataset [38], at least 5000 *birds* are oriented to the left and other 5000 oriented to the right. In the **Aircraft-100** dataset [23] with 10000 images, we can also find more than 4800 left-oriented and more than 4500 right-oriented *aircrafts*, respectively.

We perform a simple case study on the **Aircraft-100** dataset [23] to reveal how image reversal prevents us from achieving satisfying classification performance. We choose the **Aircraft-100** dataset for the reason that the orientation of an *aircraft* is more easily determined than a *bird*.

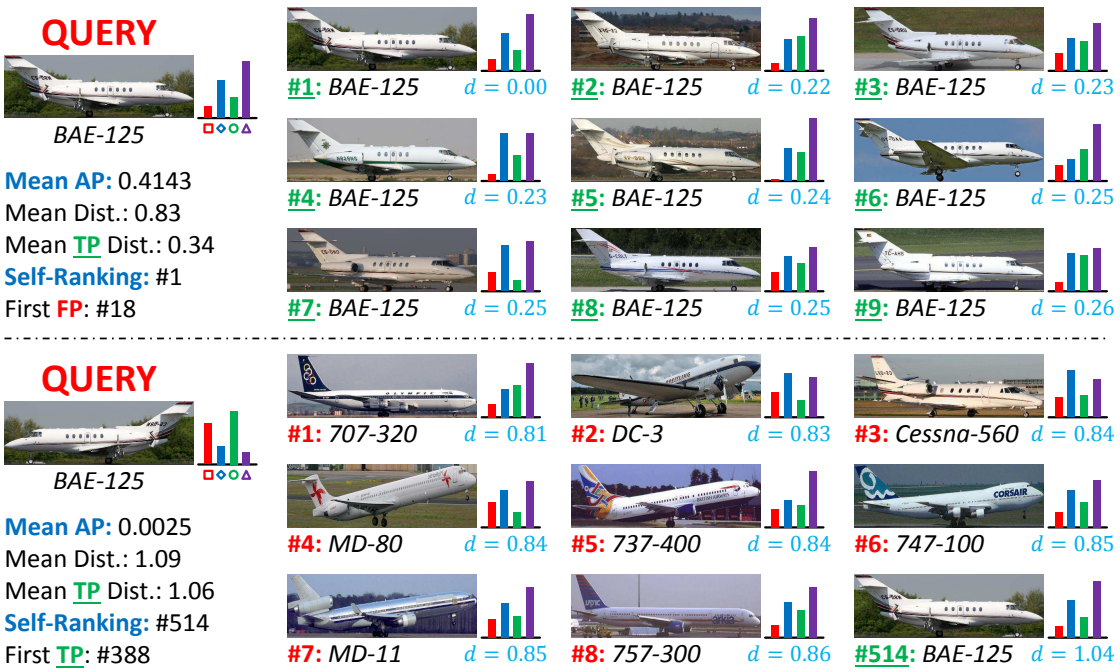

Figure 2: Retrieval in the right-oriented dataset with a right-oriented image and its reversed version (best viewed in color).

Based on the original dataset, we manually reverse the left-oriented images, generating a right-aligned dataset. We extract image features with the settings in Section 4.1, and use them for both image classification and retrieval.

With the standard training/testing split (around 67 images per category are used for training), the recognition rate is 53.13% on the original dataset and rises up quickly to 63.94% on the right-aligned dataset, with more-than-10% absolute accuracy gain (more-than-20% relative gain). This implies that orientation alignment brings a huge benefit for fine-grained object recognition.

As another interesting experiment, we use all (10000) images in the right-aligned dataset for training, and evaluate the model on two datasets with exactly the same image contents but different orientations. When testing images are all right-oriented (*i.e.*, performing self-validation), the classification accuracy is 99.73%. However, when testing images are all left-oriented (by reversing right-oriented ones), the accuracy drops dramatically to 46.84%. This experiment indicates that a model learned from right-oriented objects may not recognize left-oriented objects very well.

As an intuitive demonstration, we perform retrieval on the right-aligned dataset. We sort the candidate images according to the ℓ_2 distance between the image representation vectors. Results are summarized to in Figure 2. When the query is of the same orientation (right) with the database, the search result is satisfied (mAP is 0.4143, the first false-positive is ranked at $\#18$). However, if the query image is reversed, its feature representation changes thoroughly, and the retrieval accuracy drops dramatically (mAP is 0.025, the first true-positive is ranked at $\#388$). It is worth noting, in the latter case, that the reversed version of the query image is ranked at $\#514$, which means that more than 500 images, mostly from different categories, are more similar to the query than its reversed copy!

Since an image and its reversed copy might have totally different feature representation, in a fine-grained dataset containing both left-oriented and right-oriented objects, we are implicitly partitioning the images of each class into two (or even more) prototypes. Consequently, the number of training images of each prototype is reduced and the risk of over-fitting increased. With this observation, some algorithms [7][6] augment the dataset by generating a reversed copy for each image to increase the number of training cases of each prototype, leading to recognition accuracy gain. We propose a different idea that focuses on generating feature representation which is invariant to image reversal.

3.2. Towards Reversal Invariance

We start from observing how SIFT, a typical handcrafted descriptor, changes with left-right image reversal. The structure of a SIFT descriptor is illustrated in Figure 3. A patch is partitioned into 4×4 spatial grids, and in each grid a 8-dimensional gradient histogram is computed. Here we assume that spatial grids are traversed from top to bottom, then left to right, and gradient intensities in each grid is

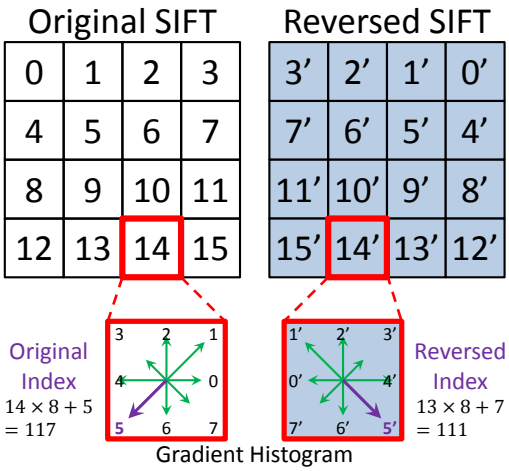

Figure 3: SIFT and its reversed version. Same number indicates corresponding grids/gradients. Numbers in original SIFT indicate the order of collecting grids/gradients.

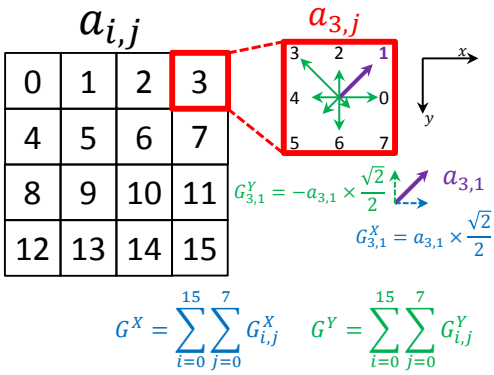

Figure 4: Estimating the orientation of SIFT.

collected in a counter-clockwise order. When an image is left-right reversed, all the patches on it are reversed as well. In a reversed patch, both the order of traversing spatial grids and collecting gradient values are changed, although the absolute gradient values in the corresponding directions do not change. Taking the lower-right grid in the original SIFT descriptor (#15) as the example. When the image is reversed, this grid appears at the lower-left position (#12), and the order of collecting gradients in the grid changes from $(0, 1, 2, 3, 4, 5, 6, 7)$ to $(4, 3, 2, 1, 0, 7, 6, 5)$.

Denote the original SIFT as $\mathbf{d} = (d_0, d_1, \ldots, d_{127})$, in which $d_{i\times 8+j} = a_{i,j}$ for $i = 0, 1, \ldots, 15$ and $j = 0, 1, \ldots, 7$. As shown in Figure 3, each index (0 to 127) of the original SIFT is mapped to another index of the reversed SIFT. For example, d_{117} ($a_{14,5}$, the bold arrow in Figure 3) would appear at d_{111} ($a_{13,7}$) when the descriptor is reversed. Denote the **index mapping function** as $f^{\mathrm{R}}(\cdot)$ (*e.g.*, $f^{\mathrm{R}}(117) = 111$), so that the reversed SIFT could be computed as: $\mathbf{d}^{\mathrm{R}} = f^{\mathrm{R}}(\mathbf{d}) = \left(d_{f^{\mathrm{R}}(0)}, d_{f^{\mathrm{R}}(1)}, \ldots, d_{f^{\mathrm{R}}(127)}\right)$.

Towards reversal invariance, we need to design a **descriptor transformation function** $r(\mathbf{d})$, which satisfies $r(\mathbf{d}) = r\left(\mathbf{d}^{\mathrm{R}}\right)$ for any descriptor $\mathbf{d}$. For this, we define $r(\mathbf{d}) = s\left(\mathbf{d}, \mathbf{d}^{\mathrm{R}}\right)$, in which $s(\cdot, \cdot)$ satisfies symmetry, *i.e.*, $s(\mathbf{d}_1, \mathbf{d}_2) = s(\mathbf{d}_2, \mathbf{d}_1)$ for any pair $(\mathbf{d}_1, \mathbf{d}_2)$. In this way reversal invariance is achieved: $r(\mathbf{d}) = s\left(\mathbf{d}, \mathbf{d}^{\mathrm{R}}\right) = s\left(\mathbf{d}^{\mathrm{R}}, \mathbf{d}\right) = s\left(\mathbf{d}^{\mathrm{R}}, \left(\mathbf{d}^{\mathrm{R}}\right)^{\mathrm{R}}\right) = r\left(\mathbf{d}^{\mathrm{R}}\right)$. We use the fact that $\left(\mathbf{d}^{\mathrm{R}}\right)^{\mathrm{R}} = \mathbf{d}$ holds for any $\mathbf{d}$.

3.3. RIDE on SIFT Descriptors

There are a lot of symmetric function $s(\cdot, \cdot)$, such as dimension-wise summation or maximization. Here we consider only one type of functions which select one of two candidates as the result, *i.e.*, $s\left(\mathbf{d}, \mathbf{d}^{\mathrm{R}}\right)$ equals to either $\mathbf{d}$ or $\mathbf{d}^{\mathrm{R}}$, so that the descriptive power of SIFT is maximally preserved. In general, we can define an **orientation quantization function** $q(\cdot)$, and choose the one in $\left\{\mathbf{d}, \mathbf{d}^{\mathrm{R}}\right\}$ with the larger function value. Ideally, $q(\cdot)$ could capture the orientation property of a descriptor, *e.g.*, $q(\mathbf{d})$ reflects the extent that $\mathbf{d}$ is oriented to the right. Recall that in the original version of SIFT [21], each descriptor is naturally assigned an orientation angle $\theta \in [0, 2\pi)$, so that we can simply take $q(\mathbf{d}) = \cos\theta$, but orientation is often ignored in the implementation of dense SIFT [4][37]. We aim at recovering the orientation with fast computations.

The major conclusion is that, the global orientation of a densely-sampled SIFT descriptor could be estimated by its local gradients. For each of the 128 dimensions, we take its gradient value and lookup for its (1 of 8) direction. The gradient value is then decomposed into two components along the x-axis and y-axis, respectively. The left/right orientation of the descriptor is then computed by collecting the x-axis components over all the 128 dimensions. Formally we define 8 orientation vectors $\mathbf{u}_j$, $j = 0, 1, \ldots, 7$. According to the definition of SIFT in Figure 3, we have $\mathbf{u}_j = \left(\cos(j\pi/8), \sin(j\pi/8)\right)^\top$. The global gradient could be computed as $\mathbf{G}(\mathbf{d}) = (G_x, G_y)^\top = \sum_{i=0}^{15}\sum_{j=0}^{7} a_{i,j}\mathbf{u}_j$. The computing process is illustrated in Figure 4. The proof of estimation could be found in the supplementary material.

We may simply take G_x as the value of quantization function, *i.e.*, $q(\mathbf{d}) = G_x(\mathbf{d})$ for every $\mathbf{d}$. It is worth noting that $q(\mathbf{d}) = -q\left(\mathbf{d}^{\mathrm{R}}\right)$ holds for any $\mathbf{d}$, therefore we can simply use the sign of $q(\mathbf{d})$ to compute the reversal invariant descriptor transform $\widetilde{\mathbf{d}}$:

$$\widetilde{\mathbf{d}} = r(\mathbf{d}) = \begin{cases} \mathbf{d} & q(\mathbf{d}) > 0 \\ \mathbf{d}^{\mathrm{R}} & q(\mathbf{d}) < 0 \\ \max\left\{\mathbf{d}, \mathbf{d}^{\mathrm{R}}\right\} & q(\mathbf{d}) = 0 \end{cases} \quad (1)$$

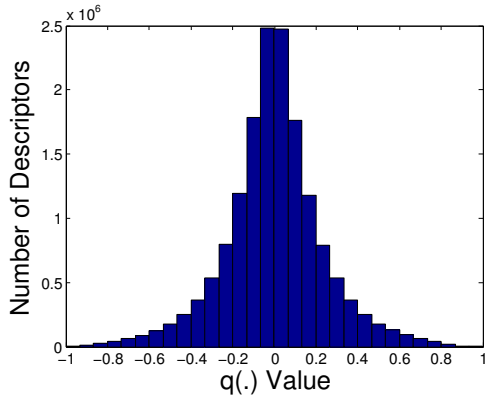

Figure 5: The distribution of $q(\cdot)$ values on the **Bird-200** dataset. According to the implementation of [37], SIFT descriptors are ℓ_2-normalized so that $\|\cdot\|_2 = 1$.

where $\max\{\mathbf{d}, \mathbf{d}^{\mathrm{R}}\}$ denotes either $\mathbf{d}$ or $\mathbf{d}^{\mathrm{R}}$ with the larger sequential lexicographic order [46]. We name the algorithm **RIDE** (Reversal Invariant Descriptor Enhancement).

Regarding stability, RIDE could also suffer from numerical stability issues especially in areas with low gradient magnitudes. Meanwhile, when the quantization function value $q(\mathbf{d})$ is close to 0, $r(\mathbf{d})$ could be sensitive to small image noises, which may change the sign of $q(\mathbf{d})$. To quantitatively analyze the impact of image noises, we first estimate the distribution of $q(\mathbf{d})$ on the **Bird-200** dataset [38]. According to the histogram in Figure 5, one may observe that about half of SIFT descriptors have relatively small $q(\cdot)$ values. We then add a random Gaussian noise with standard deviation 0.1203 (the median of $|q(\cdot)|$ values) to each of the $q(\cdot)$ value, and find that random noises only cause the classification accuracy of SIFT case drop by less than 1%, which is relatively smaller compared to the gain of RIDE (6.37%, see Table 1(d)). Similar experiments on the **Aircraft-100** dataset [23] also lead to the same result.

3.4. Generalization

We generalize RIDE for (a) other local descriptors and (b) more types of reversal invariance.

When RIDE is applied on other descriptors, we can first extract SIFT descriptors on the same patches, then compute $\mathbf{G}$ to estimate the orientation of those patches, and perform reversal operation if necessary. A generalized flowchart of RIDE is illustrated in Algorithm 1. The extra time overheads in this process mainly come from the computation of SIFT, which sometimes could be saved with a quick recovery of SIFT from given descriptors. For example, RGB-SIFT is composed of three SIFT vectors $\mathbf{d}_{\mathrm{R}}$, $\mathbf{d}_{\mathrm{G}}$ and $\mathbf{d}_{\mathrm{B}}$, from individual red, green and blue channels, therefore we can compute $\mathbf{G}_{\mathrm{R}}$, $\mathbf{G}_{\mathrm{G}}$ and $\mathbf{G}_{\mathrm{B}}$ individually, and combine

Algorithm 1 **R**eversal **I**nvariant **D**escriptor **E**nhancement

1: **Input:** $\mathcal{D} = \{\mathbf{d}_m, \mathbf{l}_m\}_{m=1}^{M}$.
2: **procedure** RIDE
3: **Reversal:** $\mathcal{D}^{\mathrm{R}} = \{\mathbf{d}_m^{\mathrm{R}}, \mathbf{l}_m\}_{m=1}^{M}$;
4: **SIFT:** $\mathcal{D}^{\mathrm{S}} = \{\mathbf{d}_m^{\mathrm{S}}, \mathbf{l}_m\}_{m=1}^{M}$, if necessary;
5: **Orientation:** $q(\mathbf{d}_m) = G_x(\mathbf{d}_m^{\mathrm{S}})$;
6: **Selection:** $\widetilde{\mathbf{d}_m} = r(\mathbf{d}_m)$, based on (1).
7: **end procedure**
8: **Output:** $\widetilde{\mathcal{D}} = \left\{\widetilde{\mathbf{d}_m}, \mathbf{l}_m\right\}_{m=1}^{M}$.

them with $\mathbf{G} = 0.30\mathbf{G}_{\mathrm{R}} + 0.59\mathbf{G}_{\mathrm{G}} + 0.11\mathbf{G}_{\mathrm{B}}$. For other color SIFT descriptors, the only difference lies in the linear combination coefficients. By this trick we can perform RIDE on Color-SIFT descriptors very fast.

In the case that RIDE is applied on fast binary descriptors for image retrieval, we could also use the same idea to design the gradient $\mathbf{G}$ without computing SIFT. Let us take the BRIEF descriptor [5] as an example. For a BRIEF descriptor $\mathbf{d}$, $G_x(\mathbf{d})$ is obtained by accumulating the *binary tests*. For each tested pixel pair (p_1, p_2) with distinct x-coordinates, if the left pixel has a smaller intensity value, add 1 to $G_x(\mathbf{d})$, otherwise subtract 1 from $G_x(\mathbf{d})$. If x-coordinates of p_1 and p_2 are the same, this pair is ignored. $G_x(\mathbf{d})$ is similarly computed, and $q(\mathbf{d}) = G_x(\mathbf{d})$ as usual. This idea could also be generalized to other binary descriptors such as ORB [32], which is based on BRIEF.

RIDE is also capable of cancelling out a larger family of reversal operations, including the upside-down image reversal, and image rotation by $90°$, $180°$ and $270°$. For this we need to constrain the descriptor more strictly with global gradient $\mathbf{G} = (G_x, G_y)^{\top}$. Recall that limiting $G_x > 0$ selects 1 descriptor from 2 candidates, resulting in **RIDE-2** (equivalent to **RIDE** mentioned previously) for left-right reversal invariance. Similarly, limiting $G_x > 0$ and $G_y > 0$ selects 1 from 4 descriptors, obtaining **RIDE-4** for both left-right and upside-down reversal invariance, and limiting $G_x > G_y > 0$ obtains **RIDE-8** for both reversal and rotation invariance. We do not use RIDE-4 and RIDE-8 in this paper, since upside-down reversal and heavy rotations are not often observed in fine-grained datasets, and the descriptive power of a descriptor is reduced by strong constraints on $\mathbf{G}$. An experimental analysis of RIDE-4 and RIDE-8 could be found in the supplementary material.

3.5. Application on Image Classification

Finally, we briefly discuss the application of RIDE for image classification. Consider an image $\mathbf{I}$, and a set of, say, SIFT descriptors extracted from the image: $\mathcal{D} = \{\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_M\}$. When the image is left-right reversed, the set $\mathcal{D}$ becomes: $\mathcal{D}^{\mathrm{R}} = \{\mathbf{d}_1^{\mathrm{R}}, \mathbf{d}_2^{\mathrm{R}}, \ldots, \mathbf{d}_M^{\mathrm{R}}\}$. If the de-

scriptors are not reversal invariant, *i.e.*, $\mathcal{D} \neq \mathcal{D}^{\mathrm{R}}$, the feature representation produced by $\mathcal{D}$ and $\mathcal{D}^{\mathrm{R}}$ might be totally different. With RIDE, we have $\widetilde{\mathbf{d}} = \widetilde{\mathbf{d}^{\mathrm{R}}}$ for any $\mathbf{d}$, therefore $\widetilde{\mathcal{D}}$ and $\widetilde{\mathcal{D}^{\mathrm{R}}}$ are identical. Consequently, we generate the same representation for an image and its reversed copy.

A simple trick applies when RIDE is adopted with Spatial Pyramid Matching [19]. Note that corresponding descriptors might have different x-coordinates on an image and its reversed copy, *e.g.*, a descriptor appearing at the upper-left corner of the original image could also be found at the upper-right corner of the reversed image, resulting in the difference in spatial pooling bin assignment. To cope with, we can count the number of descriptors reversed by RIDE, *i.e.*, the ones satisfying $\widetilde{\mathbf{d}} \neq \mathbf{d}$. If the number is larger than $M/2$, where M is the number of descriptors, we left-right reverse the descriptor set by replacing the x-coordinate of each descriptor with $W - x$, where W is the image width. This is equivalent to predicting the orientation of an image using orientation statistics on local descriptors.

3.6. Comparison with Previous Works

To the best of our knowledge, many recently published papers achieve reversal invariance with dataset augmentation [39][7][6][28]. We will demonstrate in Section 4.2 that RIDE works better than dataset augmentation.

Although some reversal invariant descriptors have been proposed for image retrieval [15][22][55][46], these descriptors have not been adopted in classification tasks. We implement MI-SIFT [22] and Max-SIFT [46], and compare them with RIDE in Table 2 (see Section 4.2). One can observe that RIDE significantly outperforms MI-SIFT and Max-SIFT in every single case. Especially, MI-SIFT works even worse than original descriptors, which is probably because it destroys the spatial structure of SIFT and thus harms the descriptive power of SIFT.

4. Experiments

4.1. Datasets and Settings

We evaluate our algorithm on four publicly available fine-grained object recognition datasets, *i.e.*, the Oxford **Pet-37** dataset [27] (37 cat/dog breeds, 7349 images), the **Aircraft-100** dataset [23] (100 aircraft models, 100 images for each model), the Oxford **Flower-102** dataset [26] (8189 flower images from 102 categories) and the Caltech-UCSD **Bird-200**-2011 dataset [38] (11788 bird images over 200 different species). For the **Aircraft-100** and **Bird-200** datasets, a bounding box is provided for each image. The numbers of training images per category for the above datasets are about 100, 20, 67 and 30, respectively.

Basic experimental settings follow the recent proposed BoF model [33]. An image is scaled, with the aspect ratio preserved, so that there are 300 pixels on the larger axis. We only use the region within the bounding box if it is available. We use VLFeat [37] to extract dense RootSIFT [2] descriptors. The spatial stride and window size of dense sampling are 6 and 12, respectively. On the same set of patches, LCS, RGB-SIFT and Opponent-SIFT descriptors are also extracted. RIDE is thereafter computed for each type of descriptors. The dimensions of SIFT, LCS and color SIFT descriptors are reduced by PCA to 64, 64 and 128, respectively. We cluster the descriptors with a GMM of 32 components, and use the improved Fisher vectors (IFV) for feature encoding. A spatial pyramid with 4 regions (the entire image and three horizontal stripes) is adopted. Features generated by SIFT and LCS descriptors are concatenated as the FUSED feature. The final vectors are square-root normalized followed by ℓ_2 normalized [18], and then fed into LibLINEAR [10], a scalable SVM implementation. Averaged accuracy over all the categories are reported on the fixed training/testing split provided by the authors.

To compare our results with the state-of-the-art classification results, **strong** features are extracted by resizing the images to 600 pixels in the larger axis, using spatial stride 8, window size 16, and 256 GMM components.

4.2. Object Recognition

We report fine-grained object recognition accuracy with different descriptors in Table 1. Beyond original descriptors, we adopt both RIDE and dataset augmentation. By augmentation we mean to generate a reversed copy for each training/testing image, use the enlarged set to train the model, test with both original and reversed samples, and predict the label with a soft-max function [28].

In Table 1, one can see that **RIDE** produces consistent accuracy gain beyond original descriptors (**ORIG**). Moreover, when we use SIFT or Color-SIFT descriptors, **RIDE** also produces higher accuracy than that using dataset augmentation (**AUGM**). When the LCS descriptors are used, **RIDE** works a little worse than **AUGM**, which is probably because the orientation of LCS (not a gradient-based descriptor) is not very well estimated with SIFT gradients.

We shall emphasize that dataset augmentation (**AUGM**) requires almost doubled computational costs than those of **RIDE** (see Section 4.5 for details), since the time/memory complexity of many classification models is proportional to the number of training/testing images. To make fair comparison, we double the codebook size used in **RIDE** to obtain longer features, since it is a common knowledge that larger codebook sizes often lead to better classification results. Such system, denoted by **RIDE**$\times$**2**, works consistently better than **AUGM** in every single case.

We also use strong features and compare **RIDE** with other reversal invariant descriptors, namely MI-SIFT [22], Max-SIFT [46], FIND [15] and F-SIFT [55]. We compute these competitors for each SIFT component in RGB-

	ORIG	RIDE	AUGM	RIDE×2
SIFT	37.92	42.28	42.24	**45.61**
LCS	43.25	44.27	45.12	**46.83**
FUSED	52.06	54.69	54.67	**57.51**
RGB-SIFT	44.90	47.35	46.98	**49.53**
OPP-SIFT	46.53	49.01	48.72	**51.19**

(a) **Pet-37** Performance

	ORIG	RIDE	AUGM	RIDE×2
SIFT	53.13	57.82	57.16	**60.14**
LCS	41.82	42.86	43.13	**44.81**
FUSED	57.36	61.27	60.59	**63.62**
RGB-SIFT	57.89	63.09	62.48	**65.11**
OPP-SIFT	47.06	53.12	51.39	**55.79**

(b) **Aircraft-100** Performance

	ORIG	RIDE	AUGM	RIDE×2
SIFT	53.68	59.12	58.01	**61.09**
LCS	73.47	75.30	75.88	**77.40**
FUSED	76.96	80.51	79.49	**82.14**
RGB-SIFT	71.52	74.97	74.18	**77.10**
OPP-SIFT	76.12	79.68	78.83	**81.69**

(c) **Flower-102** Performance

	ORIG	RIDE	AUGM	RIDE×2
SIFT	25.77	32.14	31.60	**34.07**
LCS	36.18	38.50	38.97	**40.16**
FUSED	38.11	44.73	43.98	**46.38**
RGB-SIFT	31.36	39.16	38.79	**41.73**
OPP-SIFT	35.40	42.18	41.72	**44.30**

(d) **Bird-200** Performance

Table 1: Classification accuracy (%) of different models: using SIFT, LCS, FUSED(SIFT and LCS features concatenated) and Color-SIFT features, with RIDE or dataset augmentation. Here, **ORIG** and **RIDE** denote using original descriptors without and with RIDE, **AUGM** is for dataset augmentation, and **RIDE×2** for RIDE with doubled codebook size.

	ORIG	RIDE	MI[22]	Max[46]	FIND[15]	F[55]	[1]	[23]	[25]	[28]	[30]	[41]
P-37	60.24	**63.49**	58.91	60.65	59.63	61.06	54.30	–	56.8	–	–	59.29
A-100	74.61	**78.92**	72.26	74.39	74.06	75.95	–	48.69	–	–	–	–
F-102	83.53	**86.45**	81.06	83.13	82.91	84.72	80.66	–	84.6	–	–	75.26
B-200	47.61	**50.81**	45.59	47.20	47.49	48.21	–	–	33.3	45.2	44.2	–

Table 2: Classification accuracy (%) comparison with recent works. We use RGB-SIFT on the **Aircraft-100** dataset, and FUSED (SIFT with LCS) features on other datasets. We implement MI-SIFT [22] and Max-SIFT [46] ourselves.

SIFT, and leave LCS unchanged in FUSED. Results are shown in Table 2. The consistent 3%-4% gain verifies that RIDE makes stable contribution to visual recognition. Moreover, researchers design complex part-based recognition algorithms on the **Bird-200** dataset [6][12][44] [52][54][51][20]. We also evaluate RIDE on the detected parts provided by symbiotic segmentation and localization [6] and gravitational alignment [12]. RIDE boosts the recognition accuracy of [6] and [12] from 56.6% to 60.7% and from 65.3% to 67.4%, respectively. In comparison, [12] applies dataset augmentation to boost the accuracy from 65.3% to 67.0%. RIDE produces better results with only half time/memory consumptions. With the parts learned by deep CNNs [51], we get 73.1% with FUSED features.

4.3. Global Reversal vs. Local Reversal

Based on the above experiments, one can conclude that RIDE produces powerful image features and cooperates with detected object parts for fine-grained recognition.

An essential difference between RIDE and dataset augmentation comes from the comparison of local and global image reversal. By local reversal we mean that RIDE can decide whether to reverse every single descriptor individually, while dataset augmentation only allows to choose one image from two candidates, *i.e.*, either original or globally reversed. Figure 6 compares both strategies in an intuitive manner. In these cases, we aim at matching a **target** image with a possibly reversed **test** image. With global reversal, we have only two choices and the flexibility of our model is limited. With local reversal, however, it is possible to reverse smaller regions such as the turned *head* of the *bird* or *cat*. By this we can find larger numbers of true feature matches and obtain more similar image representation (smaller feature distances). Therefore, it is not difficult to understand the reason why RIDE works even better than dataset augmentation.

4.4. Other Recognition Results

To reveal that RIDE could also be used on other classification tasks, we perform experiments on the **LandUse-21**

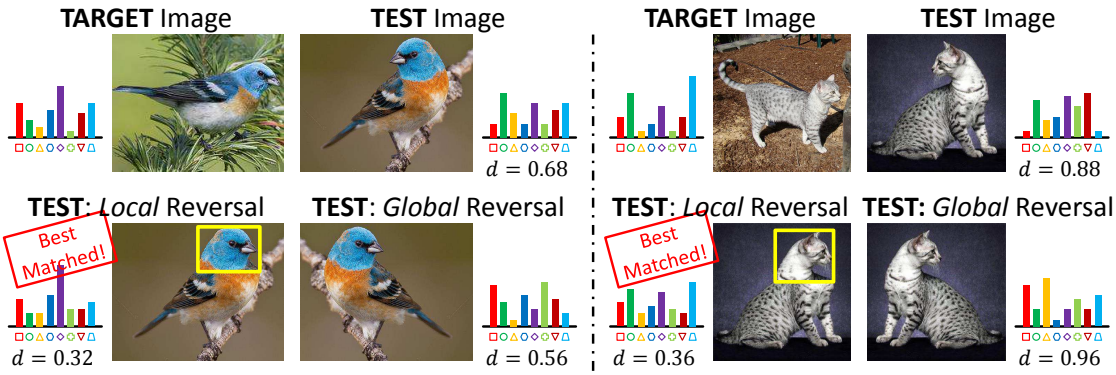

Figure 6: *Global* vs. *local* image reversal. Local reversal (with manually labeled regions in the yellow boxes) allows more flexible image representation, and produces smaller feature distances between **test** images and **target** images.

	L-21	I-67	S-397	Caltech256
ORIG	93.64	63.17	48.35	58.77
RIDE	**94.71**	**64.93**	**50.12**	**60.25**
[16]	–	63.10	–	–
[17]	92.8	63.4	46.1	57.4
[48]	–	63.48	45.91	–
[18]	–	–	49.5	–

Table 3: Classification accuracy (%) comparison with recent works on scene recognition datasets.

	ORIG	RIDE	AUGM	RIDE×2
Descriptor	2.27 hrs	2.29 hrs	2.30 hrs	2.29 hrs
Codebook	0.13 hrs	0.13 hrs	0.13 hrs	0.27 hrs
Encoding	0.78 hrs	0.78 hrs	1.56 hrs	1.28 hrs
Recognition	1.21 hrs	1.21 hrs	2.46 hrs	2.42 hrs
(RAM cost)	3.71 GB	3.71 GB	7.52 GB	7.51 GB

Table 4: Time/memory cost in each step of the BoF model. All the data are recorded with SIFT descriptors with 32 GMM components on the **Bird-200** dataset [38].

dataset [50], the MIT **Indoor-67** dataset [31], the **SUN-397** dataset [42] and the Caltech256 dataset [14]. FUSED (SIFT with LCS) features are extracted with RIDE, and results are summarized in Table 3. It is interesting to see that RIDE also works well to outperform the recent competitors.

The success on other recognition tasks indicates that reversal invariance is a common requirement in recognition. Although RIDE is motivated by the observation on fine-grained cases, it enjoys good recognition performance on a wide range of image datasets.

4.5. Computational Costs

We report the time/memory cost of RIDE with SIFT in Table 4. Since the only extra computation comes from gradient accumulation and descriptor permutation, the additional time cost of RIDE is merely about 1% of SIFT computation. RIDE does not require any extra memory storage. However, if the dataset is augmented with left-right image reversal, one needs to compute and store two instances for each image, descriptor and feature vector, resulting in almost doubled time and memory overheads, which is comparable with using a double-sized codebook, whereas the latter produces better classification results.

5. Conclusions and Future Works

In this paper, we propose **RIDE** (Reversal Invariant Descriptor Enhancement) which brings reversal invariance to local descriptors. Our idea is inspired by the observation that most handcrafted descriptors are not reversal invariant, whereas many fine-grained datasets contain objects with different left/right orientations. RIDE cancels out the impact of image/object reversal by estimating the orientation of each descriptor, and then forcing all the descriptors to have the same orientation. Experiments reveal that RIDE significantly improves the accuracy of fine-grained object recognition and scene classification with very few computational costs. RIDE is robust to small image noises. Compared with dataset augmentation, RIDE produces better results with lower time/memory consumptions.

In the future, we will generalize RIDE to more computer vision algorithms. As stated in Section 3.4, we can apply RIDE to image retrieval with fast binary descriptors. Moreover, RIDE could also inspire other related algorithms based on deep learning. For example, it is possible to design reversal invariant pooling neurons in CNNs for better performance, and the variation of RIDE might also help to improve other image representation models such as [43].

References

[1] A. Angelova and S. Zhu. Efficient Object Detection and Segmentation for Fine-Grained Recognition. *CVPR*, 2013.

[2] R. Arandjelovic and A. Zisserman. Three Things Everyone Should Know to Improve Object Retrieval. *CVPR*, 2012.

[3] T. Berg and P. Belhumeur. POOF: Part-based One-vs-One Features for Fine-Grained Categorization, Face Verification, and Attribute Estimation. *CVPR*, 2013.

[4] A. Bosch, A. Zisserman, and X. Munoz. Scene Classification via pLSA. *ICCV*, 2006.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. *ECCV*, 2010.

[6] Y. Chai, V. Lempitsky, and A. Zisserman. Symbiotic Segmentation and Part Localization for Fine-Grained Categorization. *ICCV*, 2013.

[7] K. Chatfield, V. Lempitsky, A. Vedaldi, and A. Zisserman. The Devil is in the Details: An Evaluation of Recent Feature Encoding Methods. *BMVC*, 2011.

[8] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[9] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. *CVPR*, 2005.

[10] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin. LIBLINEAR: A Library for Large Linear Classification. *JMLR*, 2008.

[11] J. Feng, B. Ni, Q. Tian, and S. Yan. Geometric Lp-norm Feature Pooling for Image Classification. *CVPR*, 2011.

[12] E. Gavves, B. Fernando, C. Snoek, A. Smeulders, and T. Tuytelaars. Local Alignments for Fine-Grained Categorization. *IJCV*, 2014.

[13] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. *International Conference on Computer Vision*, 2:1458–1465, 2005.

[14] G. Griffin. Caltech-256 Object Category Dataset. *Technical Report, Caltech*, 2007.

[15] X. Guo and X. Cao. FIND: A Neat Flip Invariant Descriptor. *ICPR*, 2010.

[16] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that Shout: Distinctive Parts for Scene Classification. *CVPR*, 2013.

[17] T. Kobayashi. Dirichlet-based Histogram Feature Transform for Image Classification. *CVPR*, 2014.

[18] M. Lapin, B. Schiele, and M. Hein. Scalable Multitask Representation Learning for Scene Classification. *CVPR*, 2014.

[19] S. Lazebnik, C. Schmid, and J. Ponce. Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories. *CVPR*, 2006.

[20] L. Li, Y. Guo, L. Xie, X. Kong, and Q. Tian. Fine-Grained Visual Categorization with Fine-Tuned Segmentation. *ICIP*, 2015.

[21] D. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 2004.

[22] R. Ma, J. Chen, and Z. Su. MI-SIFT: Mirror and Inversion Invariant Generalization for SIFT Descriptor. *CIVR*, 2010.

[23] S. Maji, E. Rahtu, J. Kannala, M. Blaschko, and A. Vedaldi. Fine-Grained Visual Classification of Aircraft. *Technical Report*, 2013.

[24] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust Wide-Baseline Stereo from Maximally Stable Extremal Regions. *Image and Vision Computing*, 2004.

[25] N. Murray and F. Perronnin. Generalized Max Pooling. *CVPR*, 2014.

[26] M. Nilsback and A. Zisserman. Automated Flower Classification over a Large Number of Classes. *ICCVGIP*, 2008.

[27] O. Parkhi, A. Vedaldi, A. Zisserman, and C. Jawahar. Cats and Dogs. *CVPR*, 2012.

[28] M. Paulin, J. Revaud, Z. Harchaoui, F. Perronnin, and C. Schmid. Transformation Pursuit for Image Classification. *CVPR*, 2014.

[29] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher Kernel for Large-scale Image Classification. *ECCV*, 2010.

[30] J. Pu, Y. Jiang, J. Wang, and X. Xue. Which Looks Like Which: Exploring Inter-class Relationships in Fine-Grained Visual Categorization. *ECCV*, 2014.

[31] A. Quattoni and A. Torralba. Recognizing Indoor Scenes. *CVPR*, 2009.

[32] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: an Efficient Alternative to SIFT or SURF. *ICCV*, 2011.

[33] J. Sanchez, F. Perronnin, T. Mensink, and J. Verbeek. Image Classification with the Fisher Vector: Theory and Practice. *IJCV*, 2013.

[34] L. Skelly and S. Sclaroff. Improved Feature Descriptors for 3D Surface Matching. 2007.

[35] T. Tuytelaars. Dense Interest Points. *CVPR*, 2010.

[36] K. Van De Sande, T. Gevers, and C. Snoek. Evaluating Color Descriptors for Object and Scene Recognition. *IEEE TPAMI*, 2010.

[37] A. Vedaldi and B. Fulkerson. VLFeat: An Open and Portable Library of Computer Vision Algorithms. *ACMMM*, 2010.

[38] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. *Technical Report*, 2011.

[39] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong. Locality-Constrained Linear Coding for Image Classification. *CVPR*, 2010.

[40] Z. Wang, B. Fan, and F. Wu. Local Intensity Order Pattern for Feature Description. *ICCV*, 2011.

[41] Z. Wang, J. Feng, and S. Yan. Collaborative Linear Coding for Robust Image Classification. *IJCV*, 2014.

[42] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. SUN Database: Large-scale Scene Recognition from Abbey to Zoo. *CVPR*, 2010.

[43] L. Xie, R. Hong, B. Zhang, and Q. Tian. Image Classification and Retrieval are ONE. *ICMR*, 2015.

[44] L. Xie, Q. Tian, R. Hong, S. Yan, and B. Zhang. Hierarchical Part Matching for Fine-Grained Visual Categorization. *ICCV*, 2013.

[45] L. Xie, Q. Tian, M. Wang, and B. Zhang. Spatial Pooling of Heterogeneous Features for Image Classification. *IEEE TIP*, 2014.

[46] L. Xie, Q. Tian, and B. Zhang. Max-SIFT: Flipping Invariant Descriptors for Web Logo Search. *ICIP*, 2014.

[47] L. Xie, Q. Tian, and B. Zhang. Simple Techniques Make Sense: Feature Pooling and Normalization for Image Classification. *IEEE TCSVT*, 2015.

[48] L. Xie, J. Wang, B. Guo, B. Zhang, and Q. Tian. Orientational Pyramid Matching for Recognizing Indoor Scenes. *CVPR*, 2014.

[49] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification. *CVPR*, 2009.

[50] Y. Yang and S. Newsam. Bag-of-Visual-Words and Spatial Extensions for Land-Use Classification. *ICAGIS*, 2010.

[51] N. Zhang, J. Donahue, R. Girshick, and T. Darrell. Part-based R-CNNs for Fine-Grained Category Detection. *ECCV*, 2014.

[52] N. Zhang, R. Farrell, F. Iandola, and T. Darrell. Deformable Part Descriptors for Fine-Grained Recognition and Attribute Prediction. *ICCV*, 2013.

[53] S. Zhang, Q. Tian, G. Hua, Q. Huang, and S. Li. Descriptive Visual Words and Visual Phrases for Image Applications. *ACM International Conference on Multimedia*, 2009.

[54] X. Zhang, H. Xiong, W. Zhou, and Q. Tian. Fused One-vs-All Mid-Level Features for Fine-Grained Visual Categorization. *ACMMM*, 2014.

[55] W. Zhao and C. Ngo. Flip-Invariant SIFT for Copy and Object Detection. *IEEE TIP*, 2013.

[56] X. Zhou, K. Yu, T. Zhang, and T. Huang. Image Classification using Super-Vector Coding of Local Image Descriptors. *ECCV*, 2010.